Efficient autonomous learning for statistical pattern recognition

J. B. Hampshire II
Jet Propulsion Laboratory, M/S 238-420
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
hamps@bvd.jpl.nasa.gov

B.V.K. Vijaya Kumar
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3890
kumar@previa.ece.cmu.edu

Reprinted from the Proceedings of the 1994 SPIE International Symposium on Optical Engineering in Aerospace Sensing, vol. 2243: Applications of Artificial Neural Networks V, S. K. Rogers & D. W. Ruck, ed's., April 1994.

ABSTRACT

We describe a neural network learning algorithm that implements differential learning in a generalized backpropagation framework. The algorithm regulates model complexity during the learning procedure, generating the best low-complexity approximation to the Bayes-optimal classifier allowed by the training sample. It learns to recognize handwritten digits of the AT&T DB1 database. Learning is done with little human intervention. The algorithm generates a simple neural network classifier from the benchmark partitioning of the database; the classifier has 650 total parameters and exhibits a test sample error rate of 1.3%.

1 INTRODUCTION

Recent advances in machine learning theory make it possible to generate pattern classifiers that are consistently robust estimates of the Bayes-optimal (i. e., minimum probability-of-error) classifier. Moreover, these advances guarantee good approximations to the Bayes-optimal classifier from models with the minimum functional complexity (e. g., the fewest parameters) necessary. These findings present a challenge: by what means can the classifier consistently and autonomously learn a robust, low-complexity approximation to the Bayes-optimal classifier? We describe a neural network learning algorithm that implements differential learning in a generalized backpropagation framework. The algorithm regulates model complexity during the learning procedure, generating the best low-complexity approximation to the Bayes-optimal classifier allowed by the training sample. We focus on the algorithm's ability to learn an optical character recognition task with little human intervention. It learns to recognize handwritten digits of the AT&T DB1 database (provided by Dr. Isabelle Guyon). The algorithm generates a simple neural network classifier from the benchmark partitioning of the database; the classifier has 650 total parameters and exhibits a test sample error rate of 1.3%.

1.1 Differential learning, efficiency, and minimum complexity

As we described in this forum last year [5], differential learning is discriminative; it seeks to partition feature vector space in the Bayes-optimal fashion by optimizing a classification figure-of-merit (CFM) objective function [7, 4, 3]. CFM objective functions are best described as differentiable approximations to a counting function: they count the number of correct classifications (or, equivalently, the number of incorrect classifications) the classifier makes on the training sample. By optimizing such an objective function, differential learning generates robust approximations to the Bayes-optimal

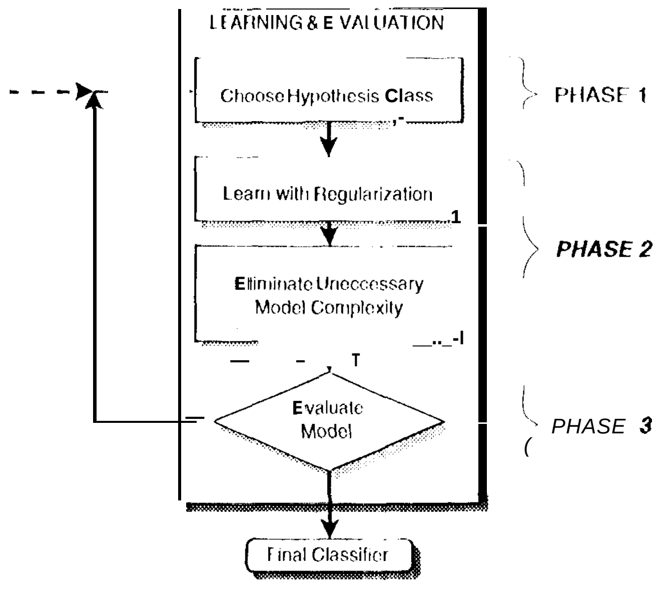


Figure 1: A diagrammatic view of an iterative, autonomous learning procedure.

classifier, generally requiring the smallest training sample size, and *always* requiring the least complex model necessary to approximate the Bayes error rate with specified precision.

When the classifier constitutes a "proper parametric model" of the data [4, ch's. 3-4], classical probabilistic learning strategies generate the best approximation to the Bayes-optimal classifier with the smallest training sample size necessary. However, when the classifier constitutes an 'improper parametric model'' of the data [4, ch's. 3-4], differential learning generates the best approximation with the smallest training sample size necessary. The most efficient learning strategy therefore depends on whether or not the classifier is a proper model of the data, if indeed such a model exists. Kolmogorov's theorem [12], can (arguably) be interpreted to mean that finding the proper parametric model for a set of stochastic concepts represented by a random feature vector is either easy or hard - there is little middle ground. Proceeding from this basis, differential learning is likely to be the most efficient choice of strategy for scenarios in which a proper model is not obvious.

1.2 Automating the learning process

Having chosen differential learning, we are still faced with selecting a model with which to generate a classifier from our training data. The minimum-complexity requirements of differential learning ensure that whatever our choice of *hypothesis class* (i. e., the model's functional basis — linear, logistic linear, or radial basis function, to name a few), we will need the least complex model in that class (e.g., the one with the fewest parameters) necessary for Bayesian discrimination. Much of the work required to find that minimum-complexity model can be done by the differential learning procedure itself. In that spirit, we describe an implementation of differential learning that automates much of phases 2 and 3 of the learning and classifier evaluation procedures, as diagrammed in figure 1. In section 3 we describe how the algorithm regulates its own learning rate, how it regulates model complexity through regularization and parameter elimination, and how it regulates the level of detail that can be learned from the training sample patterns. In section 4 we describe the statistical tests that are generated by the algorithm, both during and after learning, and we describe how these tests are used to evaluate the learned model. We conclude with a brief description of our ongoing efforts to build a truly autonomous differential learning algorithm.

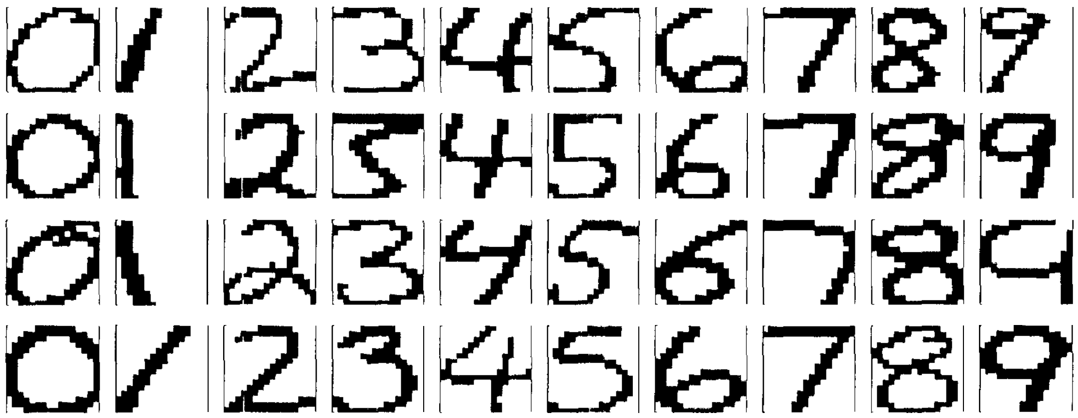


Figure 2: Forty digits randomly chosen from the AT&T DB1 database.

2 DATA SET AND EXPERIMENTAL PROTOCOL

We begin with a description of the learning/pattern recognition task that we use to illustrate our algorithm. The AT&T DB1 database contains 1200 handwritten digits: ten examples of each digit, obtained from each of twelve different subjects [2]. Figure 2 illustrates 40 examples from the database. Each example is a 256-pixel (16 x 16) binary image (i.e., pixels are either black or white). We compress each example to a 64 pixel (8 x 8 , 5-levels/pixel) image: the value of each compressed pixel is simply the average of its four constituent uncompressed pixels. Even after compression, the examples are well-defined to the human eye and have uniform scale and orientation (see [5]). Since its introduction, the database has become a benchmark standard for evaluating learning procedures and neural network architectures in the optical character recognition (OCR) domain. We in turn use the database to illustrate our learning algorithm.

Throughout this paper we refer to a "benchmark split" of the DB1 database. This term refers to the partitioning of the database into a training, sample and test sample. Both samples contain 600" examples. The benchmark training sample comprises the first five examples of each digit, obtained from each of the twelve subjects. The benchmark test sample comprises the last five examples of each digit, obtained from each of the twelve subjects. This split of the data has been used in a number of previous papers on the database; we use it so the reader can compare our results with previously published ones. The experiments described in this paper are part of a larger set ics of learning and recognition experiments using the database. These experiments are described in [5] and [4, ch. 8].

3 AUTOMATED LEARNING

As diagrammed in figure 1, the learning and model evaluation process begins with model selection. In the current implementation of our learning algorithm, the classifier model is selected by the human operator, not by the algorithm itself. That is, the choice of hypothesis class (selection of functional basis and interconnection topology) is determined by the operator *a priori*. Learning begins after the model has been selected.

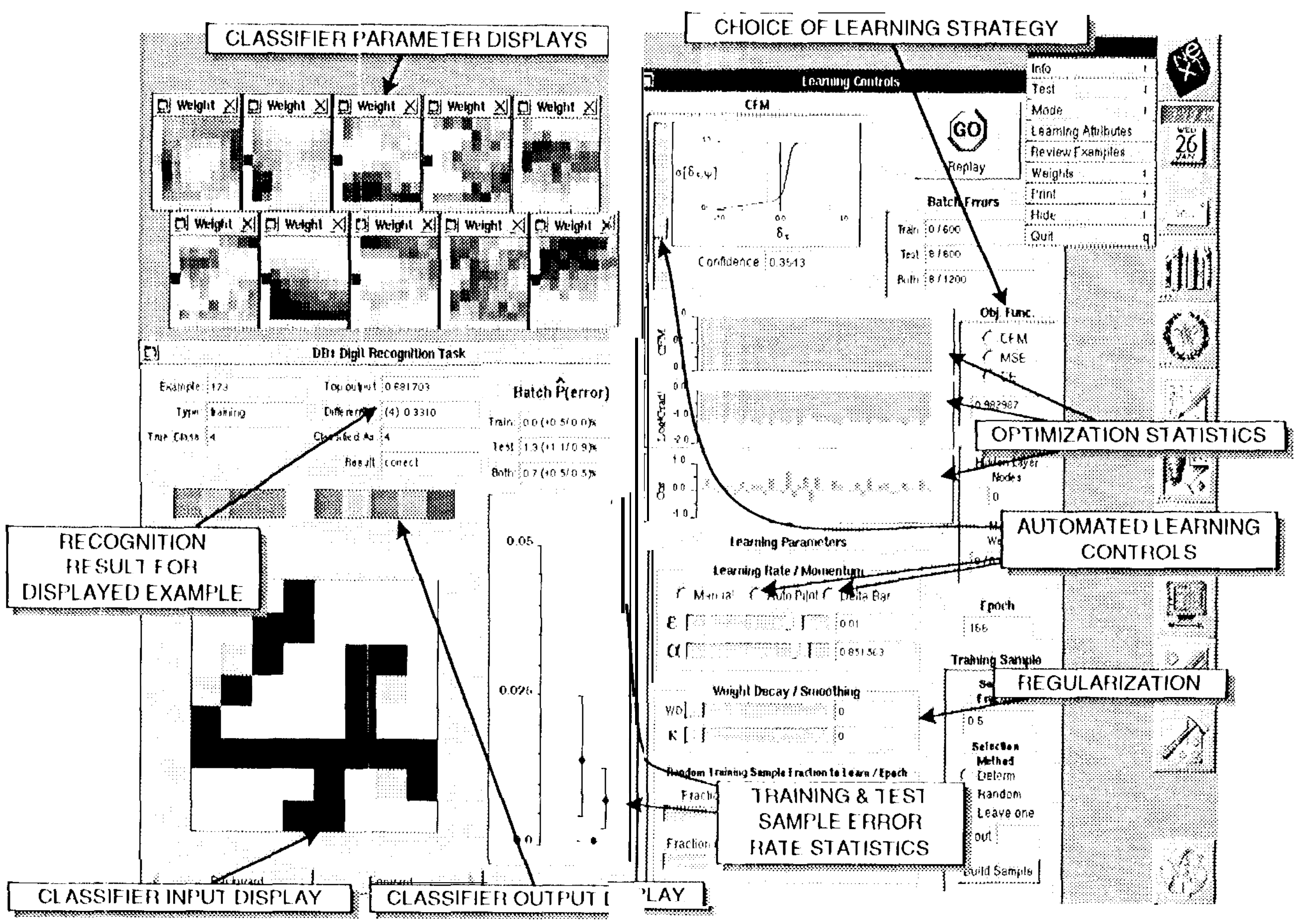


Figure 3: A full-screen image of our learning algorithm and its main displays. Classifier state and learning controls are annotated.

3.1 Review of the basic learning algorithm

Figure 3 shows a full-screen image of the learning algorithm and its main displays. The main window on the right contains all the controls and monitoring displays for learning, which we describe from top to bottom. The top display shows the form of the synthetic CFM objective function, given the present level of its learning confidence parameter (described below). The size of the training, test, and combined samples are shown in a form to the right of the CFM display; this form also shows the current number of training errors during learning, and the test and combined error counts during testing. The choice of learning strategy is determined by the choice of objective function used for learning. CFM, mean-squared error (MSE), and the Kullback-Leibler information distance [13] (a.k.a. cross entropy) objective functions can be optimized. CFM implements differential learning; the other two error measures implement probabilistic learning. The three displays to the left show the value of the objective function; the $\log_{10}$ magnitude of the objective function's gradient on parameter space at the current parameter vector value; the cosine of the angle between the current gradient and the previous iteration's gradient. This last metric is commonly used to show whether or not consecutive parameter vector changes are in the same direction: if the cosine is near unity, consecutive steps are in the same direction, and the search is not oscillating in parameter space; if the cosine is near zero, consecutive steps are orthogonal; if the cosine is negative the search is oscillating in parameter space. A sequence of the last 50 sets of these three statistics is shown. Controls for the learning rate and momentum terms for gradient ascent/descent (ϵ and α respectively see (1) in appendix 1) are below the optimization statistics. These controls are initialized to default values of .01 and .85, but are generally controlled by the learning algorithm itself during learning (see below). Below these are controls for two forms of *regularization*: weight decay (e. g., [8]) and weight smoothing. Weight smoothing constrains parameters corresponding to local neighborhoods in the retinotopic feature vector to have similar values. Our implementation of both regularization procedures is detailed in [4, appendix M]. Controls for sub-sampling the training sample during each learning epoch are at the bottom of the control panel. Controls for partitioning data into training and test samples are to the right of these.

The main window on the left of the display shows the classifier state: we describe it from top to bottom. The top-left form describes the input pattern of the classifier. The middle right form describes how the classifier has recognized the input pattern. The far-right form shows recognition error rates for the training sample during learning; during testing this form also shows the test and combined sample error rates. Whisker plots corresponding to these error rates are shown below this form: these plots have 95% confidence bounds. The classifier's input pattern is shown in the large pixel map display on the lower left. A pixel display of the classifier's output state is shown immediately above the input display. Hidden layer nodes (not used in the classifier displayed) are shown between the input and output displays. Mouse-clicking on any node pops up a graphical display of all the weights (i.e., parameters) feeding into that node. The parameters connecting the classifier input to each of the 10 output nodes are shown in the top-left of the screen. The parameters corresponding to the nodes that represent "0" "4" form the top row of these displays; parameters corresponding to the nodes that represent "5" "9" form the bottom row. Abstract representations of each digit are evident in these displays.

3.2 Automated learning functions

Four principal control functions of learning need to be regulated

- learning rate
- model complexity (via regularization)
- learning confidence (i.e., the level of detail in the feature vector that can be learned from the training sample: the lower the confidence one requires of the classifier, the greater the detail it can learn from the training sample)
- termination of learning

We have automated three of these four control functions in our algorithm. Regularization is not controlled directly in this implementation; the human operator must set the level of weight decay and/or weight smoothing prior to learning. We discuss the automated controls in the following paragraphs.

3.2.1 Learning rate automation

The learning rate and momentum terms for gradient ascent/descent are controlled manually by default. The learning rate can be controlled automatically using one of two schemes: "auto-pilot" or *modified Delta-Bar-Delta*. Owing to its superiority, we describe only the latter control method, which is a variation of the Delta-Bar-Delta procedure described in [10, 18, 1, 11]. Details of our variant are given in appendix A.

Typically one learning rate ϵ is used for all the parameters of the classifier, and gradient ascent/descent proceeds according to (1) in appendix A. References [10, 18, 1, 11] describe why this can be sub-optimal and propose that each parameter have its own learning rate. Each rate is increased when the current parameter update and an exponential average of previous updates are of the same sign; when the current and average of previous updates have different signs an indication of oscillation in the search algorithm due to an excessive learning rate the learning rate is reduced. The procedure leads t(I faster convergence of the **learning** algorithm's search for optimal parameters, but learning rates tend to oscillate, due to the first **order** autoregressive nature **of** the learning rate control ' 'filter''.

Wc change the control **filler's** characteristic **so** *that it* **uses a** *more stable moving average* **paradigm** that operates **on** both short **and** long time **scales (again, scc appendix A).** The resulting learning rates respond **quickly** to **changes** in the objective function's local topology on parameter **space,** but they do not oscillate.

3.2,2 Sc heduled reduction of **learning confidence**

When differential learning **is** conducted, the CFM confidence parameter **plays** an important role in determining the level **of** discriminative **detail** that can be learned from the feature vectors compri **sing** the **[raining,** sample. When the synthetic CFM confidence parameter **is** high, *the objective function* **is a** *very* **weak** *(nearly* **linear)** approximation to **a** correct **classification** counting function. **As** con fidence **is** reduced, **the** objective function becomes **a** better approximation to a step function that counts correct **classifications.** By maximizing **this** objective function, we maximize **the** number of correct classifications the **classifier makes on the** training sample. The **lower** the confidence, **the more likely the classifier will** be **able** to learn **all of** the trai ni ng examples.

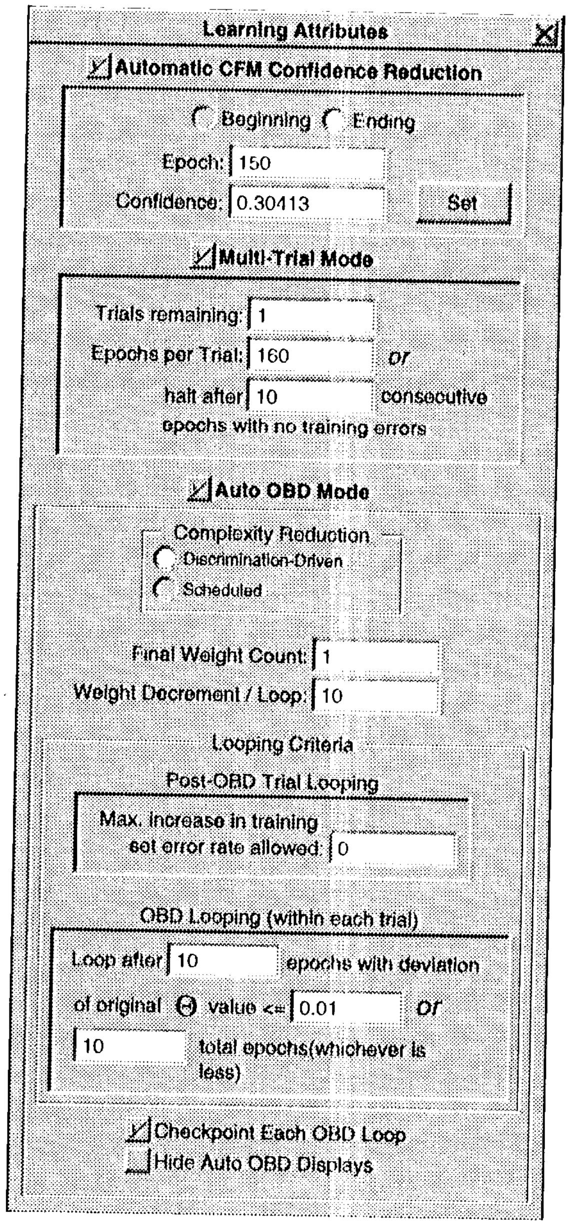
A rigorous theoretical **motivation** for the synthetic CFM objective function's confidence parameter **is given** in **[4,** ch's. **2** *&* **71.** Simply put, **easy examples,** which lie near the modes of the feature vector's class-conditional probability density functions (pdfs), can be learned with high confidence (i.e., without attention to detail), but **hard examples ,** which lie near the **tails of** the feature vector's class-conditional pdfs near the class boundar **ics** must be learned with low confidence **(i.e.,** with increased attention to detail). **As** a result, there **is a** strong theoretical *motivation for learning with* initially high confidence, gradually reducing confidence as lea **I** ning prog resses: the **classifier** learns the **easy** training **examples** first, and then learns the **hard** ones **(see** [4, ch. **7]).** Figure 4 (top) shows controls that **allow** the human operator to select this **kind** of linear reduction schedule for the CFM confidence parameter.

On the positive side, this scheduled reduction **of** learning confidence **has a** statistically significant effect on the classifier's ability to learn (and subsequently classify) hard examples. **on** the negative **side,** the human **oi>ci-ale]** must specify **the** rate of *and* iterative *bounds on* the reduction. This inevitably induces an ad-hoc element: learning with too little con fidence gives **even classifiers** with low complexity the functional capacity to learn details that are not representative. **As a** result, we **arc** currently working on **a** truly-autonomous, decision-directed procedure for placing lower bounds on the level of confidence with which a given training sample should be lear **tied,**

3.2.3 Automatic **termination** of learning

Termination **is** generally **viewed as** an important issue **iii** connectionist learning, an unhappy consequence **of** inefficient probabilistically-generated classifiers. These classifiers often require excessive functional complexity in order to learn the training sample: the added complexity reduces discriminant bias, but the reduced bias **is** more than **offset** by an increase **in** discriminant variance **(scc** [4, ch's. 3-4] and [5]). **As a** result, the classifiers generalize poorly. Through empirical observation, numerous authors have found that early termination of **(tic** learning procedure limits the increase **in** *discriminant* variance, **so** the **classifiers** generalize better. But the scheme requires that the human operator decide when to terminate the *learning without* offering **any** objective measure **of** when to terminate.

Since differential **learning is** asymptotically efficient and since it requires the minimum functional complexity necessary to learn the training sample (see **[4,** ch. 3] **and** [6] for formal proofs), learning can proceed **as** long **as** the synthetic CFM objective function's gradient **is** non-zero. **As a** result, **wc** employ **a** simple scheme **for** automatic termination of learning (see figure 4, middle). **Wc** simply have **the lc:it-sing** algorithm halt after **a** large consecutive number of iterations **for** which the the training sample error rate **is** zero, *or* after **a** fixed number of learning iterations have transpired. By setting both numbers to **a** conservatively high number, **wc** ensure that learning proceeds until **(here is nothing** more to learn, so to speak.



Learning Attributes

✓ Automatic CFM Confidence Reduction

Beginning   Ending

Epoch: 150

Confidence: 0.30413   Set

✓ Multi-Trial Mode

Trials remaining: 1

Epochs per Trial: 160 *or*

halt after 10 consecutive epochs with no training errors

✓ Auto OBD Mode

Complexity Reduction

Discrimination-Driven

Scheduled

Final Weight Count: 1

Weight Decrement / Loop: 10

Looping Criteria

Post-OBD Trial Looping

Max. increase in training set error rate allowed: 0

OBD Looping (within each trial)

Loop after 10 epochs with deviation of original Θ value <= 0.01 *or* 10 total epochs(whichever is less)

✓ Checkpoint Each OBD Loop

Hide Auto OBD Displays

Figure 4: Controls for automated learning. From top to bottom: scheduled reduction of learning confidence; automated multi-trial learning; automated OBD (differential and probabilistic).

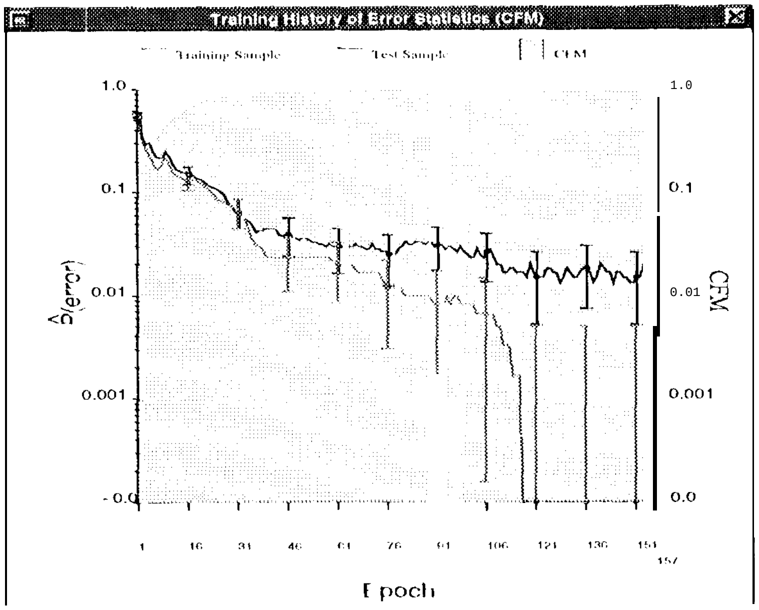


Figure 5: The empirical error rates (training sample in gray and test sample in black) for the 650-parameter logistic linear classifier as it learns the benchmark training sample differentially. The classifier's empirical test sample error rate is 1.3 (+ 1.1/-0.9)% after 160 learning epochs

3.3 OCR illustration of automated learning

Figure 5 shows both the DB 1 benchmark training sample (gray) and test sample (black) empirical error rates as differential learning progresses through approximately 160 learning epochs, using the automated procedures described above. These plots are commonly known as learning curves. A logistic linear classifier (e.g., see [5, pg. 90]) is generated differentially; learning confidence is reduced from a value of ~ 0.5 at epoch zero to $\sim$ 0.35 beyond epoch 100; learning proceeds to 160 epochs, or 50 consecutive epochs for which the training sample error rate is zero; there is no weight decay; the weight smoothing coefficient is $\kappa \cong 0.13$. The objective function's value is plotted as a light gray background in the figure. Ninety-five percent confidence intervals on the error rates are plotted at periodic intervals. From these curves, one can see that the training sample error rate is representative of the test sample error rate up to ninety differential learning epochs. Beyond this point the empirical training sample error rate is significantly lower than the test sample error rate.

There are four scatter plots on reduced *discriminator output space* in figure 6. in each plot, the final output slate of the classifier for each example in the DB 1 database is shown as a point among the scatter. Training examples are shown as light-gray dots, and test examples are shown as dark gray triangles. The four plots shown correspond to fear points (epochs) along the learning curve in figure 5: epochs 15, 30, 50, and 160. The value of the classifier output corresponding to the correct classification of an example is the abscissa (we denote this "correct" classifier output by y_T). The value of the largest classifier output corresponding to an incorrect classification of the example is the ordinate. (we denote this largest "incorrect" classifier output by $\bar{y}_T$). The *reduced discriminant boundary* is the line that separates the half of this two-dimensional space that corresponds to a correct classification (i.e., $y_T > \bar{y}_T$) from the half that represents an incorrect classification (i.e., $y_T \leq \bar{y}_T$).

At 15 epochs all the output states are clustered such that about 15% of all examples are misclassified; at 30 epochs the error rate is approximately 7%; al 50 epochs it's 2.5% for the training sample and 4.5% for the test sample; at 160 epochs all the training examples are correctly classified and 1.3% of the test examples are misclassified. Contours of constant CFM are

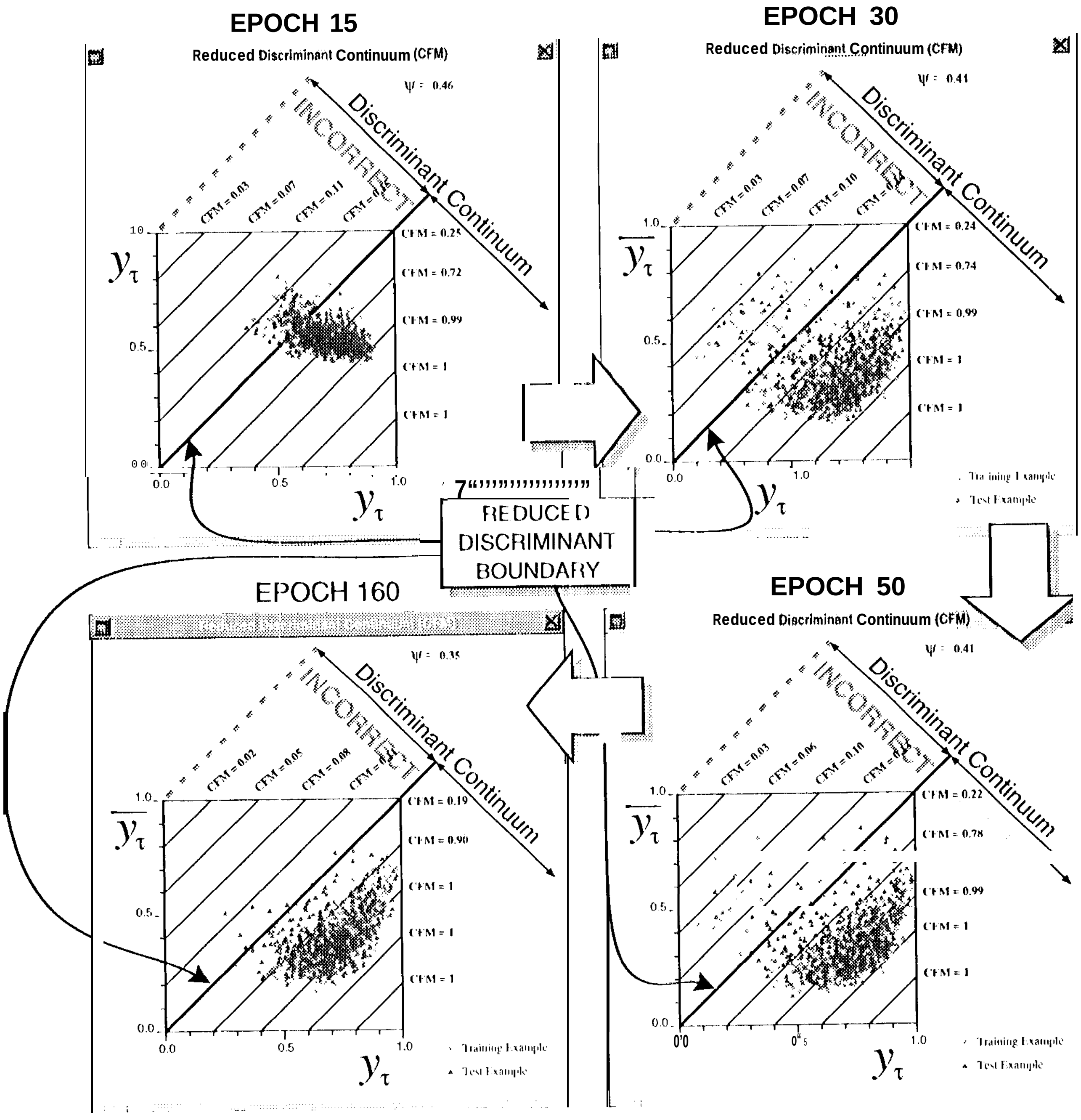


Figure 6: A sequence of four reduced discriminator output states for the 650-parameter logistic linear classifier as it learns the benchmark data split. Clockwise from top left: state at epoch 15, epoch 30, epoch 50, and epoch 160. These displays are generated automatically upon request.

shown as lines parallel to the reduced discriminant boundary on each of the four plots. Note that all the training examples lie parallel to the CFM = 0.90 contour at the end of learning (epoch 160), as do most of the test examples. The remaining test examples — ones that are hard for the classifier to recognize — fall close to the reduced discriminant boundary. Owing to the monotonic nature of the CFM objective function, these examples also lie parallel to the reduced discriminant boundary, and most of them are on the correct side of the boundary. Eight test examples lie on the boundary itself or on the incorrect side of the boundary.

These scatter plots illustrate that the differentially-generated classifier generalizes well, since the output state for the test sample is similar to the output state of the training sample. There is sufficient difference to account for a statistically significant difference between the training and test sample error rates though, a fact that is born-out by the learning curves in figure 5 at epoch 160.

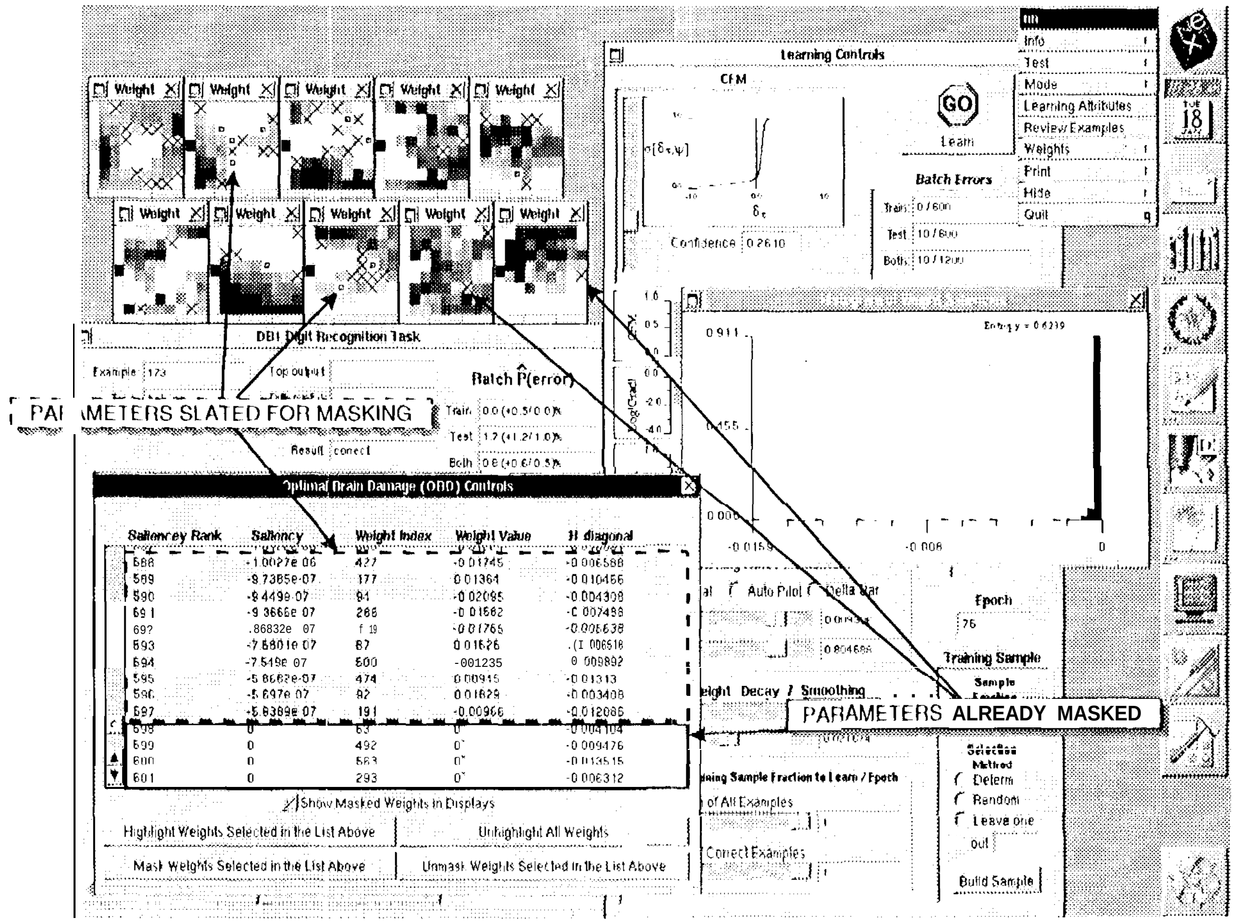
3.3.1 Autonomous differential OBD

Figure 7 shows a full-screen display of our learning algorithm as it automatically eliminates parameters from the classifier after the fully-parameterized learning just described terminates. The parameter elimination procedure is a differential variant of the OBD algorithm [14]; the variant is detailed in appendix B. The rationale behind parameter elimination is as follows: parameters that are not necessary for robust classification of the feature vector constitute excessive functional complexity, which can lead to poor generalization on the test sample; therefore, remove these parameters from the model.

Differential OBD differs from the original because it uses the synthetic CFM objective function instead of the MSE objective function, so it's linked with differential learning rather than probabilistic learning. Also, our implementation of differential OBD is an automatic procedure; no human oversight is required. Figure 4 (bottom) shows the constraints on the autonomous differential OBD procedure. The procedure begins at the termination of fully-parameterized learning; at each iteration the n (10 in this case) least salient parameters (i.e., those n that contribute least to maximizing CFM over the training sample) are eliminated or "masked"; following the masking of these parameters, the classifier learns for a user-specified number of epochs (10 in this case) or until the value of the CFM objective function climbs back to within a user-specified amount (.01 in this case) of its value prior to the parameter masking, whichever is less; if, after this learning, the training sample error rate has not risen above its pre-OBD value, OBD continues. If the error rate has risen above its pre-OBD value, the parameters last masked are restored and only $n/2$ are masked. The number of masked parameters is reduced in this manner until the iterative masking/re-learning procedure succeeds without increasing the classifier's training sample error rate above the pre-OBD rate. When no more parameters can be eliminated without increasing the training sample error rate, differential OBD terminates.

Figure 7 shows autonomous differential OBD in its early stages. Parameters that have already been eliminated and those that are about to be eliminated are shown in the weight displays. A ranked list of parameter saliencies (see appendix B) identifies the parameters slated for masking (shaded in gray). Figure 8 shows the final set of parameters after differential OBD terminates. Forty percent of the 650 original parameters have been eliminated, leaving some of the digit parameter maps quite sparse (e. g., those for "0" and "1"). Despite this large reduction in parameters, the training sample is still classified without error. A comparison of figure 9 and figure 6 (bottom left) shows that the reduced discriminator output states before and after OBD are not appreciably different. This is also evident from a comparison of figures 5 and 10: The test sample error rate has increased from 1.3% prior to OBD to 3.3% after OBD, not quite a statistically significant increase.

Though not statistically significant, the increase in the test sample error rate is a negative outcome, since it represents a decrease in the classifier's ability to generalize well, OBD after all is supposed to *improve* generalization, not degrade it. We attribute the degradation to the high degree of spatial correlation in the parameters of the classifier induced by weight smoothing during fully-parameterized learning. Eliminating parameters decorrelates the remaining parameters owing to the way the smoothing works. The decorrelation results in an unrepresentative information gain in the classifier that is sufficient to increase the test sample error rate. Thus, weight elimination can have negative results in at least some cases involving strong local correlations among the feature vector elements.


PARAMETERS SLATED FOR MASKING
PARAMETERS ALREADY MASKED
Learning Controls
CFM
GO
Learn
Batch Errors
DBI Digit Recognition Task
Batch P(error)
Optimal Brain Damage (OBD) Controls
Info
Test
Mode
Learning Attributes
Review Examples
Weights
Print
Hide
Quit
Epoch
Training Sample
Show Masked Weights in Displays
Highlight Weights Selected in the List Above
Unhighlight All Weights
Mask Weights Selected in the List Above
Unmask Weights Selected in the List Above
Build Sample

Saliencey Rank	Saliency	Weight Index	Weight Value	H diagonal
588	+1.0027e 06	427	-0 01745	-0 006588
589	-9.7385e-07	177	0 01364	-0 010466
590	-9 4499 07	94	-0 02095	-0.004308
59 1	-9 3666e 07	266	-0 01682	-C 007498
59?	.86832e 07	f 19	-0 01765	-0.006638
593	-7.6801e 07	87	0 01626	.(I 006918
594	-7.549e 07	600	-001235	0 009892
595	-5 8662e-07	474	0 00945	-0 01313
596	-5 697e 07	92	0 01829	-0 003408
597	-5.8389e 07	191	-0.00966	-0.012086
598	0	63	0	-0 004104
599	0	492	0	-0 009476
600	0	563	0	-0 013515
601	0	293	0	-0 006312

Figure 7: A full-screen image of our learning algorithm pet'fol"rllilly autonomous differential OBD. The parameters that have been masked (i.e., eliminated) are shown as black "X" S on a gray background; parameters currently selected for masking are highlighted in the OBD ranked saliency list (gray highlight) and in the parameter (or "weight") displays (outlined in white). The histogram of the classi fi er's parameters (right, middle of display) reveals that most have small values in a narrow range.

A classifier generated from a control learning/OBD experiment in which all characteristics are identical except for the learning strategy (probabilistic learning via the Kullback-Leibler information distance is substituted for differential learning via synthetic CFM) results in a post-OBD classifier with 315 masked parameters (49% of the original 650); the classifier's training sample error rate is 5.3%, and its test sample error rate is 13.2%. 'l'bus, differential OBD is more efficient than probabilistic OBD for the same reasons that differential learning, is generally more efficient than probabilistic learning.

4 AUTOMATED CLASSIFIER EVAI .UATION

Our reduced discriminator output state scatter plots arc directly related to classical receiver operator characteristic (ROC) curves for the classifier. '1'0 see how and wily, we return to the final state Of the classifier at the termination of fully-parameterized learning (epoch 160). Figure 11 shows this plot with the synthetic CFM objective function superimposed perpendicular to the reduced discriminant boundary. This perpendicular axis is known as the discriminant continuum, the

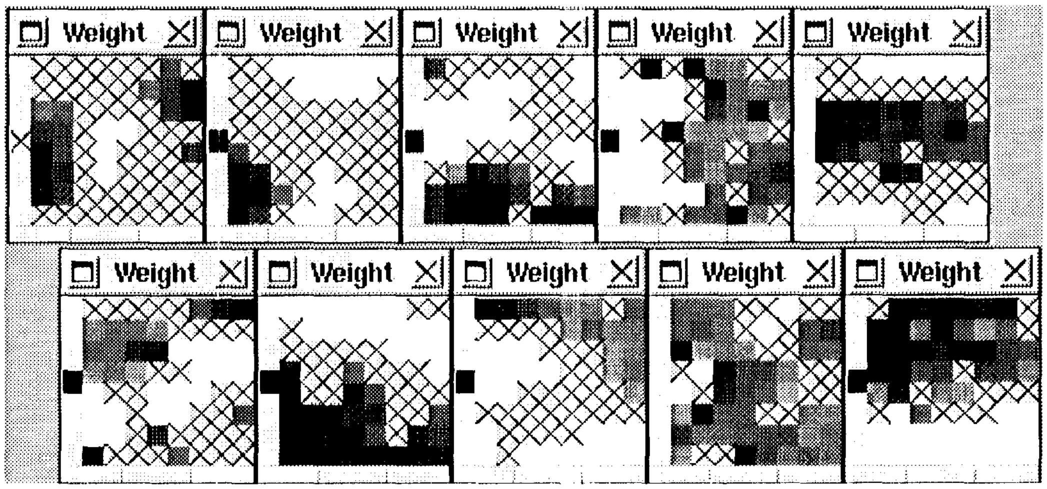
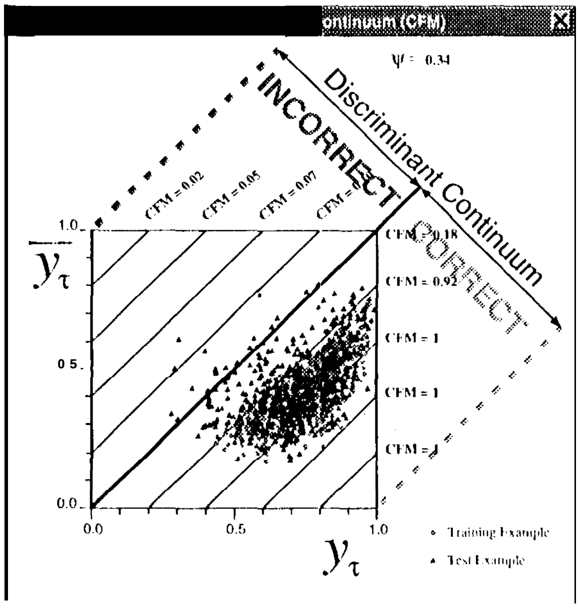


Figure 8: The final parameter state after differential OBD automatically terminates. The top row are the parameters for the digits 0 - 4; the bottom row are the parameters for the digits 5 - 9. Note in particular the sparse nature of the parameter maps for the digits 0 and 1.



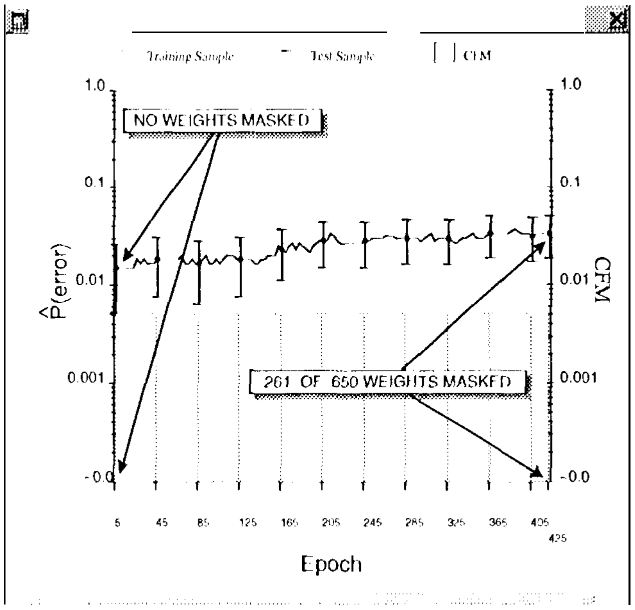


Figure 9: The final reduced discriminator output state after differential OBD automatically terminates. Note that this state is not appreciably different from the one at epoch 160 in figure 6, the classifier state just prior to beginning differential OBD. The similarity confirms that the test sample error rates before and after differential OBD are not statistically significant (cf. figures 10 and 5).

Figure 10: The training (gray) and test (black) sample error rate histories during differential OBD. Note that the test sample error rate increase is not statistically significant (cf. figure 5), even though 40% of the classifier's 650 parameters have been masked (i.e., eliminated).

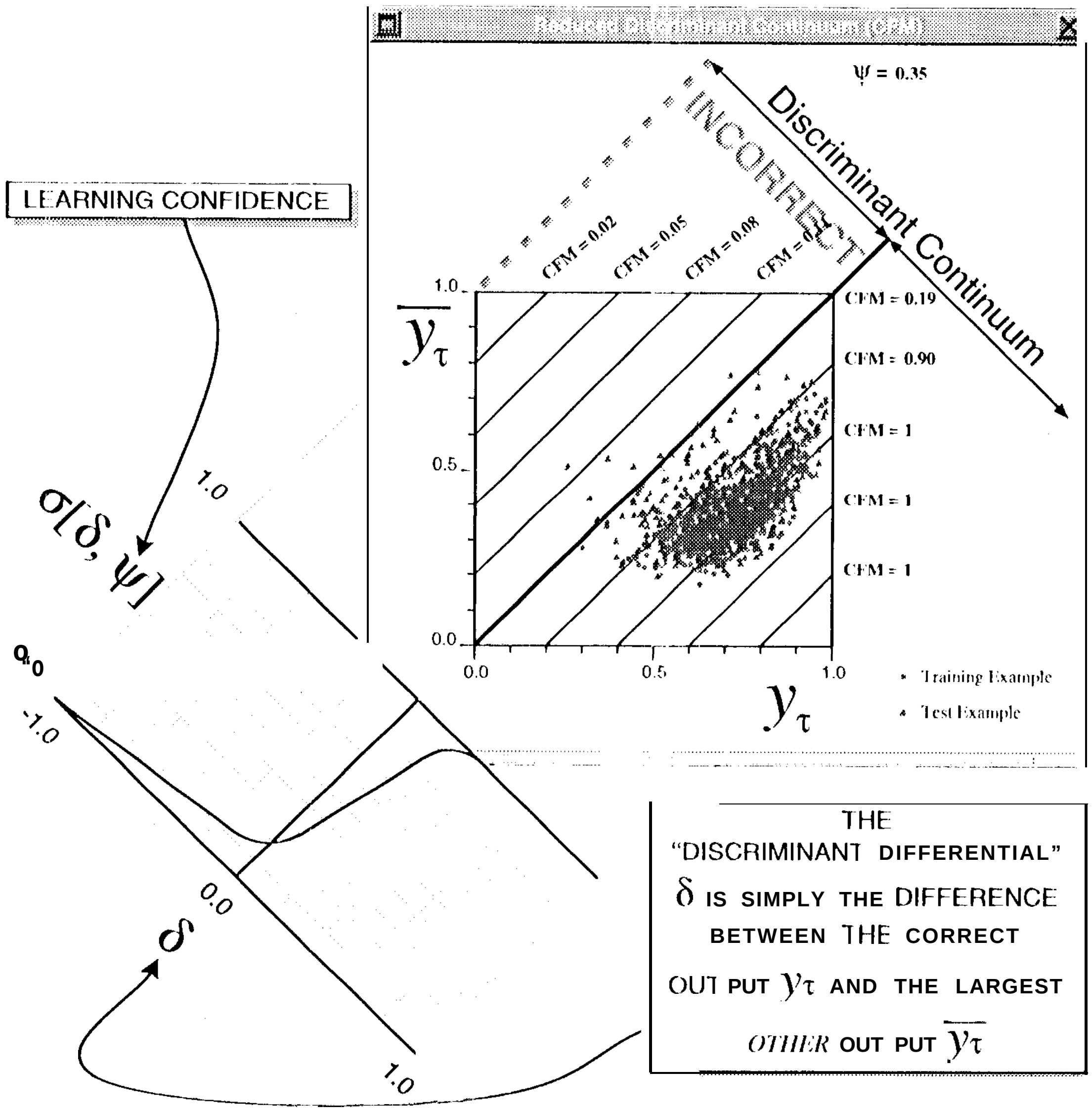
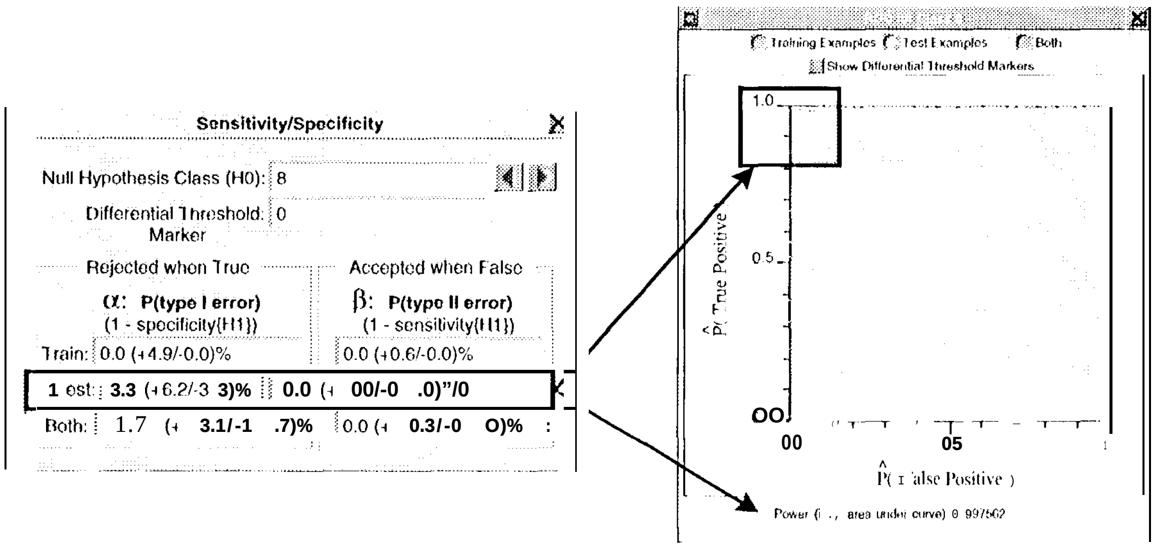


Figure 11: The reduced discriminator output state of figure 6 (epoch 160) with the synthetic CFM objective function superimposed to illustrate its relationship to the output state and the resulting *discriminant differential* $\delta = y_\tau - \overline{y_\tau}$. Negative differentials correspond to classification errors; positive ones correspond to correct classifications. The contours of constant CFM are parallel to the reduced discriminant boundary a necessary condition for efficient learning (see [4, ch. 5]).



Sensitivity/Specificity

Null Hypothesis Class (H0): 8

Differential Threshold Marker: 0

	Rejected when True	Accepted when False
	α: P(type I error) (1 - specificity{H1})	β: P(type II error) (1 - sensitivity{H1})
Train:	0.0 (+4.9/-0.0)%	0.0 (+0.6/-0.0)%
1 est:	3.3 (+6.2/-3 3)%	0.0 (+ 00/-0 .0)"/0
Both:	1.7 (+ 3.1/-1 .7)%	0.0 (+ 0.3/-0 O)%

Figure 12: Scnsilivily/specificity analyses and the receiver operator characteristic (ROC) carve for the digit "8" Of the benchmark split. These displays are generated automatically.

domain of the *discriminant differential* δ, which is simply the. difference between the classifier output (y_T) representing the correct class and the output ($\bar{y}_T$) representing largest *other* (incorrect) class. When δ is positive, the example is correctly classified; when it isn't the example is misclassified. The value of CFM increases with the value of δ, which is why maximizing CFM maximizes the number of correct classifications. Decreasing the confidence parameter ψ increases the steepness of the CFM function, making it a better approximation to a step function that simply counts correct classifications, '1'1111s, even "hard" examples with small positive values Of δ generate the maximum value Of CFM.

Altering the detection threshold for a class is accomplished simply by altering the value of δ above which an example is recognized as a member of the class. 1 has, ROC cur vcs arc easily generated by changing the position of the reduced discriminant boundary along the discriminant continuum. When the boundary is moved towards very negative values Of δ for a given class, more and more examples are recognized as members Of the class: the true positive detection rate for the class increases, but so does the false positive rate. 1 ,ikewise, as the reduced discriminant boundary is moved towards very positive values 01 δ for a given class, fewer and fewer examples are recognized as members Of the class: the true positive detection rate for the class decreases, as does the false positive rate. This is a graphical description Of the computations our algorithm performs to generate ROC carves automatically. Figure 1 2 shows the ROC curve for the digit "8" at the cad Or fully-parameterized learning, but prior 1001111. The carve corresponds to the reduced discriminator output state in figure 1 1. The classifier's sensitivity and specificity for detecting "8"s is shown to the left of the ROC cur vc: the numbers shown have 95% confidence bounds and correspond to the default discriminant boundary al δ = 0.

ROC curves allow us to characterize the trade-Ofl between the classifier's sensitivity and specificity on a class-by-class basis, and can be used in conjunction with the reduced discriminant continuum and a related graphical display in order to set rejection thresholds non-negative values Or δ below which examples will not be definitively classified, but will be rejected as too uncertain to be recognized with reasonable confidence. Consider the test example shown in figure 13: it is a "3" in the benchmark DB1 split, but the class fier recogni zes it as a "5". Figure 14 shows the details Or this misclassification in the context Of all other classifications (both training and test samples for the purpose of illustration). The top display shows what the histogram Of δ would look like for all the examples, given the classifier's parameterization al epoch 160, *assuming* that the correct classification is *always* '5". Under this 'always recognize 5" hypothesis, the vast majority Or examples generate negative values Or δ (that is, the classifier output corresponding to "5" is usually smaller

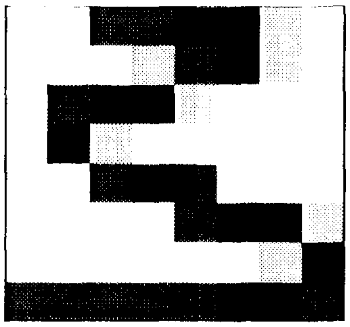


Figure 13: A "3" in the benchmark test sample that has been incorrectly classified as a "5".

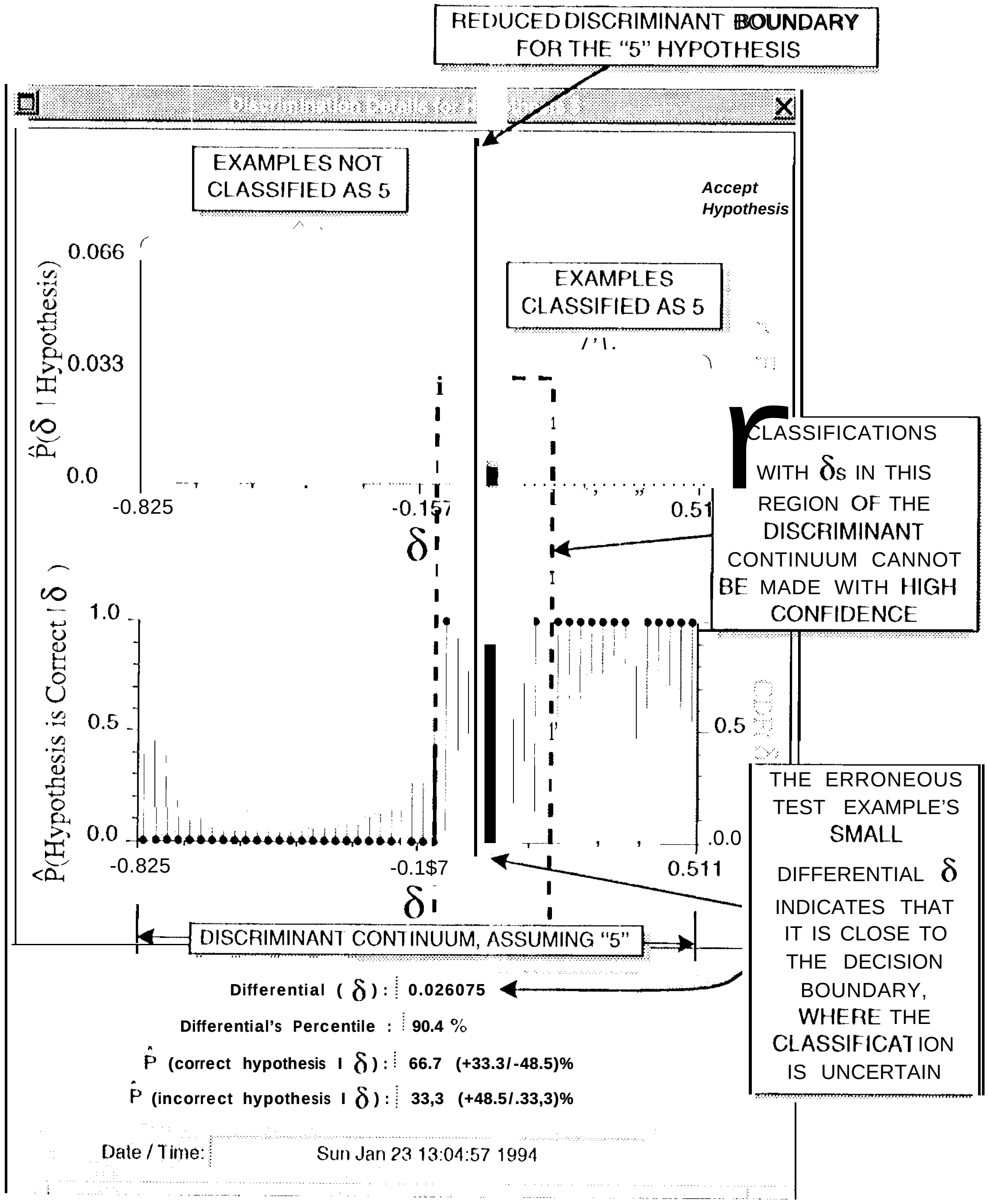
than some other output), so the hypothesis is rejected. Only about one tenth of the examples generate positive values of δ, for which the "5" hypothesis is accepted. The histogram of δ values shows that the distributions of acceptances and rejections peak relatively far from the decision threshold.

The whisker plots in the lower half of the display show the empirical probability (with 95% *confidence* bounds) that the '5" hypothesis is valid for a given value of δ. They are superimposed on a cumulative histogram that corresponds to the pdf histogram in the upper half of the display. Since few examples generate small positive and negative values of δ, the probability of a "5" hypothesis fluctuates in the vicinity of the discriminant boundary at $\delta = 0$. Moreover, the confidence bounds on the probability of a "5" hypothesis for small δ are large again, because so few examples generate these small values of δ. As a result, we cannot have reasonable confidence in a "5" classification with a value of δ that falls inside the "low confidence" region enclosed by the dashed box. That is, examples that generate a value of $\delta \lesssim .18$ cannot be classified as "5"s with confidence. The value of δ for the "3" in figure 13 is indicated by the dark-highlighted vertical bar, which is well inside the "low confidence" region. Thus, the erroneous "5" classification should be rejected. We are currently developing a statistically sound approach to automatically setting the rejection thresholds for each class based on the metrics illustrated in figure 14. At present it appears that the threshold should be set at the value of δ below which the lower 95% confidence bound on the probability of a correct classification is less than 50%.

5 CONCLUSION

We have described a learning algorithm that generates linear, multi-layer perceptron (MLP), and radial basis function (RBF) neural network classifiers with little human intervention. The algorithm adjusts its learning rates automatically, using a modified Delta-Bar-Delta procedure. The initial learning phase terminates when the training sample is classified without error or when the number of learning iterations exceeds a specified limit, whichever occurs first. The learning procedure can then initiate an automatic OBD parameter elimination procedure, which iteratively reduces classifier complexity. The procedure runs automatically, terminating in either a scheduled or a decision-directed manner. When differential learning is conducted, learning confidence is automatically reduced according to a pre-set schedule. After learning, the algorithm generates classifier sensitivity/specificity estimates, ROC curves, and learning curves. Additional automatically-generated graphical displays allow the user to analyze the classifier output state over the course of learning.

When paired with differential learning, the algorithm generates classifiers that are consistently good approximations to the Bayes-optimal classifier, as long as the initial choice of model is sufficiently complex and regularization is not excessive. At present, the model and level of regularization are chosen by the human operator prior to learning. Our current research is aimed at automating these choices as well as the post-learning evaluation described in the previous section, using an iterative procedure by which the learning algorithm generates and evaluates increasingly complex models of the


REDUCED DISCRIMINANT BOUNDARY FOR THE "5" HYPOTHESIS
EXAMPLES NOT CLASSIFIED AS 5
Accept Hypothesis
EXAMPLES CLASSIFIED AS 5
CLASSIFICATIONS WITH δs IN THIS REGION OF THE DISCRIMINANT CONTINUUM CANNOT BE MADE WITH HIGH CONFIDENCE
$\hat{P}(\delta \mid \text{Hypothesis})$
0.066
0.033
0.0
-0.825
-0.157
0.511
δ
$\hat{P}(\text{Hypothesis is Correct} \mid \delta)$
1.0
0.5
0.0
THE ERRONEOUS TEST EXAMPLE'S SMALL DIFFERENTIAL δ INDICATES THAT IT IS CLOSE TO THE DECISION BOUNDARY, WHERE THE CLASSIFICATION IS UNCERTAIN
DISCRIMINANT CONTINUUM, ASSUMING "5"

Differential (δ) : 0.026075

Differential's Percentile : 90.4 %

$\hat{P}$ (correct hypothesis | δ) : 66.7 (+33.3/-48.5)%

$\hat{P}$ (incorrect hypothesis | δ) : 33,3 (+48.5/.33,3)%

Date / Time: Sun Jan 23 13:04:57 1994

Figure 14: A detailed view of the erroneous classification of the digit "3" in figure 12: the digit is incorrectly classified as a "5". This display characterizes the "5" classification hypothesis in the context of all the test and training sample examples (see the text for explanation). Dark highlighting indicates where the classifier's output "differential" δ for the "3" example falls along the "discriminant continuum" (i.e., the domain of δ).

training sample. Such a strategy is consistent with the minimum-complexity requirements of differential learning. Our goal is a theoretically-defensible, truly autonomous differential learning machine.

6 ACKNOWLEDGMENT

This research was conducted while Dr. Hampshire was a graduate student in Carnegie Mellon University's Department of Electrical and Computer Engineering; it was supported by the Air Force Office of Scientific Research (AFOSR) *under* grant AFOSR-89-0551. Dr. Hampshire's current research is funded by the National Aeronautics and Space Administration (NASA) via the Jet Propulsion Laboratory's Office of Telecommunications and Data Acquisition (RTOP-3 10-30-72)" and by NASA's office of Advanced Concepts and Technology (Code C), Operations/AI Program.

APPENDICES

A MODIFIED DELTA-BAR-DELTA PROCEDURE

The canonical backpropagation equation [16, 17] describes how the classifier's parameter vector $\boldsymbol{\theta}$ at iteration $n+1$ is altered via a simple steepest ascent/descent algorithm

$$\boldsymbol{\theta}[n+1] = \boldsymbol{\theta}[n] + \left(\underbrace{\pm \varepsilon \cdot \nabla_{\boldsymbol{\theta}} \left(\Phi[n] \right) + \overbrace{\alpha \cdot \Delta\boldsymbol{\theta}[n-1]}^{\text{momentum term}}}_{\Delta\boldsymbol{\theta}[n]} \right) , \tag{1}$$

where the change or "deflection" in the parameter vector at time n is given by $\Delta\boldsymbol{\theta}[n]$. The sign of the *learning rate* ε depends on whether the objective function used to guide the search for optimal parameters is to be maximized (+) or minimized (-). Typically ε is fixed.

The Delta-Bar-Delta algorithm [10, 18, 11] associates a different learning rate with *each* element of the parameter vector, rather than using one rate for all the elements; furthermore, it modulates the value of each learning rate according to the rules described in [10, (4)]: rates are increased linearly and decreased exponentially based on whether or not the current weight deflection and an exponential average of past deflections are of the same sign. The scheme is effective, but can lead to oscillating learning rates, owing to its use of a first-order infinite impulse response (IIR) filter to compute the exponential average of past deflections.

We modify the Delta-Bar-Delta control filter in three ways, in order to improve its stability:

1. learning rates are increased and decreased exponentially.
2. increases in learning rates are made on a long time scale.
3. decreases in learning rates are made on both short and long time scales.

Our control filter computes long and short term averages of its input, which at iteration n is a *sign flip indicator* $s_i[n]$: a binary number that indicates whether or not the sign of the parameter deflection at time n is opposite to the sign of the deflection at time $n-1$ (we use the notation $\theta_i[n]$ to denote the ith element of the parameter vector $\boldsymbol{\theta}$ at iteration n;

likewise, we use the notation $\varepsilon_i[n]$ to denote the ith element of the associated learning rate vector ε at iteration n; the notation $\Delta\theta_i[n]$ denotes the deflection of the parameter vector's ith element at iteration n).

The learning rates are all initialized to the same value; as learning begins, each rate is then automatically adjusted by the control filter according to the following rules:

$$\varepsilon_i[n+1] = \begin{cases} .9\,\varepsilon_i[n], & \sum_{j=0}^{k-1=4} s_i[n-j] > 0 \\ .9\,\varepsilon_i[n], & \sum_{j=0}^{l-1=19} s_i[n-j] > 5 \\ 1.1\,\varepsilon_i[n], & \sum_{j=0}^{l-1=19} s_i[n-j] = 0 \\ \varepsilon_i[n], & \text{otherwise} \end{cases} \tag{2}$$

Each time ε_i is increased or decreased, the control filter's input (i, c., its history of sign flips) is *flushed,* so that ε_i is not decreased for at least another $k = 5$ iterations, nor is it increased for at least another $l = 20$ iterations. The resulting modified Delta-Bar-Delta procedure regulates learning rates quickly without making them Oscillate.

B DIFFERENTIAL OBD

Optimal Brain Damage (OBD) [14] is a second-order parameter elimination procedure; it eliminates parameters that have low "saliency" (i.e., those that have little or no effect on the objective function used to search for the classifier's optimal parameterization). The procedure assumes a diagonal hessian matrix for the second-order derivatives of the objective function with respect to the parameter vector. The Optimal Brain Surgeon (OBS) [9] algorithm assumes a full diagonal hessian matrix in order to identify the non-salient parameters more effectively. Both algorithms assume that a mean squared error (MSE) objective function is being minimized during learning.

We list the equations by which OBD/OBS can be implemented for the *arbitrary* objective function below because MSE provably leads to inefficient learning when the classifier is an improper parametric model of the data [4, ch's.3-4]. Given an improper model, the MSE-based OBD/OBS algorithms will *not* remove the least salient parameters (they will merely eliminate those parameters that contribute least to the classifier's MSE for the training sample). By pairing OBD/OBS with a synthetic form of the classification figure of merit (CFM) objective function [3], elimination of the least salient parameters (for classification purposes) is guaranteed. We refer to OBD with CFM as *differential OBD.*

We use the notation $\Phi(S^n\,|\,\theta)$ to denote the value of the arbitrary objective function Φ, given the training sample S^n and the classifier with parameterization θ. Specific expressions for the first- and second-order derivatives of Φ are left to the reader; those for the original logistic form of CFM are given in [7]; those for synthetic CFM are given in [4, appendix D]. If we know the value of Φ, given the parameter vector θ^*, we can express its value, given the parameter vector θ', using the following Taylor series expansion:

$$\begin{aligned}\Phi(S^n|\theta') &= \Phi(S^n|\theta^*) + (\theta' - \theta^*)^T\,\nabla_\theta\,(\Phi(S^n|\,\theta^*)) \\ &\quad + \tfrac{1}{2}(\theta' - \theta^*)^T\,\mathrm{H}_\theta\,(\Phi(S^n|\,\theta^*))\,(\theta' - \theta^*) + O\left[\|\theta' - \theta^*\|^3\right]\end{aligned} \tag{3}$$

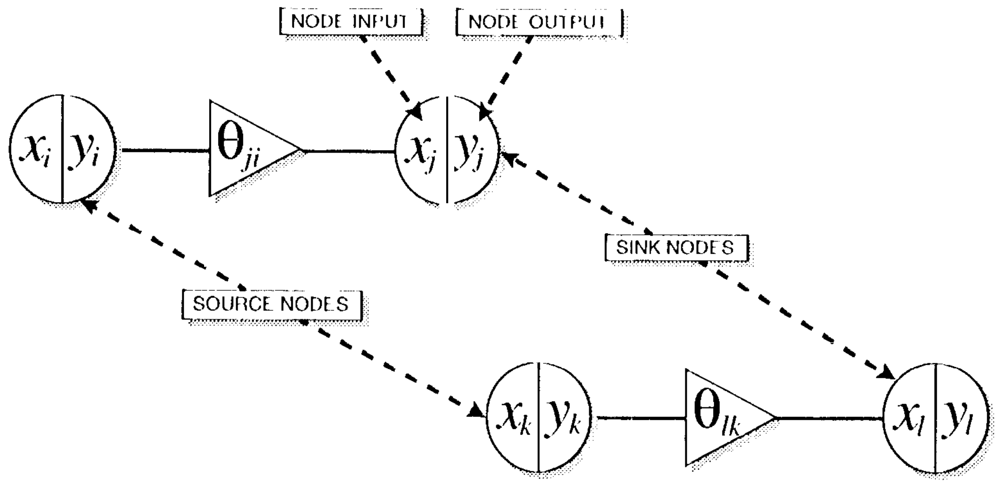


Figure 15: A diagrammatic view of two source/sink node pairs in a neural network; this view illustrates the notational convention we use in our description of the di fferential OBD algoi ithm's hessian computations. None of the four nodes is necessarily in the same layer as any other.

The notation z^T denotes the transpose of vector 2, $\nabla_\theta(\Phi(S^n|\theta^*))$ denotes the gradient of Φ with respect to the parameter vector f), evaluated at θ^*, and 110 $(\Phi(S^n|\theta^*))$ denotes the hessian of Φ with respect to the parameter vector O, evaluated al θ^*

If Φ is optimized when the classifier's parameter vector is given by θ^*, then the first-order term in (3) will be zero. Assuming that third and higher order terms arc negligible, (3) can be rearranged to form the following approximate expression for the change in the objective function's value when 0' is changed to θ' :

$$\underbrace{\Phi(S^n\,|\,\theta') - \Phi(S^n\,|\,\theta^*)}_{\Delta\Phi^*} \simeq \frac{1}{2}\underbrace{(f)' - \theta^*)^T\,\mathbf{H}_\theta\,(\Phi(S^n\,|\,\theta^*))}_{\mathcal{H}^T}(\theta' - \theta^*) \qquad (4)$$

The equations below can be used with the chain rule to compute each element Of the hessian in (4), Wc use Φ^* as short-hand for $\Phi(S^n|\theta^*)$. Figure 15 illustrates the notational conventions we use to label the nodes and parameters Of our neural network classifier: source nodes feed sink nodes via a connecting parameter or "weight". Thus, $\frac{\partial^2\Phi^*}{\partial\theta_{ji}\partial\theta_{lk}}$ denotes the j,lth element of the hessian] $\mathbf{I}_\theta(\Phi(S^n|\theta^*))$ in (1 5). It follows that the diagonal elements of the hessian arc given by

$$\frac{\partial^2\Phi^*}{\partial\theta_{ji}^2} = \frac{\partial^2\Phi^*}{\partial x_j^2}\cdot y_i^2, \qquad (5)$$

where

$$\frac{\partial^2\Phi^*}{\partial x_j^2} = \frac{\partial^2\Phi^*}{\partial y_j^2}\cdot\left(\frac{\partial y_j}{\partial x_j}\right)^2 + \frac{\partial^2 y_j}{\partial x_j^2}\cdot\frac{\partial\Phi^*}{\partial y_j}, \qquad (6)$$

and

$$\frac{\partial^2 \Phi}{\partial y_j^2} = \sum_{\lambda=1}^{\Lambda} \theta_{\lambda j}^2 \cdot \frac{\partial^2 \Phi}{\partial x_\lambda^2} \qquad (7)$$

Note that Λ is the fan-out of the source node y_j

If the full m x m element hessian is being used (on S), it is more efficient to compute the vector transpose $\mathcal{H}^T$ in (4) directly, using the procedure described by Pearlmutter [15]. The resulting computation is efficient, $O[m]$ rather than $O[m^2]$

REFERENCES

[1] A. G. Barto and R. S. Sutton. *Adaptation of Learning Rate Parameters. Goal Seeking* Components for Adaptive Intelligence: An Initial Assessment. Air Force Wright Aeronautical Laboratories/ Avionics Laboratory, Wright Patterson AFB, Ohio, 1981. Appendix C of Technical Report AFWAL-TR-81-1070."

[2] I. Guyon, V. Vapnik, B. Boser, L. Bottou, and S. Solla. Structural risk minimization for character recognition. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems, vol. 4*, pages 471--479, San Mateo, CA, 1992. Morgan Kauffman.

[3] J. B. Hampshire II. ANSI C source code for the Synthetic CFM Objective Function. Available via anonymous ftp from speech1.cs.cmu.edu (128.2.254.145): retrieve the file cfmSourceCode.c., March 1992. *The code is described fully in the author's Ph.D. thesis, op. cit.*

[4] J. B. Hampshire II. *A Differential Theory of Learning for Efficient Statistical Pattern Recognition.* PhD thesis, Carnegie Mellon University, Department of Electrical & Computer Engineering, Hammerschlag Hall, Pittsburgh, PA 15213-3890, September 1993. *An abridged version is available via anonymous ftp from* speech1.cs.cmu.edu *(128.2.254.145): retrieve the file README for directions.*

[5] J. B. Hampshire II and B. V. K. Vijaya Kumar. Differential theory of learning for efficient neural network pattern recognition. In D. Ruck, editor, *Proceedings of the 1993 SPIE International Symposium on Optical Engineering and Photonics in Aerospace and Remote Sensing, vol. 1966: Science of Artificial Neural Networks*, pages 76--95, April 1993. *The proceedings version did not print clearly. A clean copy is vailable via anonymous ftp from* speech1.cs.cmu.edu *(128.2.254.145): retrieve the file* detailedOverview.ps.Z.

[6] J. B. Hampshire II and B. V. K. Vijaya Kumar. Differentially Generated Neural Network Classifiers are Efficient. in C. A. Kamm, G. M. Kuhn, B. Yoon, R. Chellappa, and S. Y. Kung, editors, *Neural Networks for Signal Processing III: Proceedings of the 1993 IEEE Workshop*, pages 151--160, New York, September 1993. The Institute of Electrical and Electronic Engineers, inc.

[7] J. B. Hampshire II and A. H. Waibel. A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Neural Networks*, 1(2):216-228, June 1990.

[8] S. J. Hanson and L. Y. Pratt. Comparing Biases for Minimal Network Construction with Back-Propagation. In Dave Touretzky, editor, *Advances in Neural Information Processing Systems, vol. 1, pages 177--185.* Morgan Kaufmann, San Diego, CA, 1989.

[9] B. Hassibi and D. G. Stork. Second-order derivatives for network pruning: Optimal brain surgeon. in S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems, vol. 5*, pages 164--171. Morgan Kauffman, San Mateo, CA, 1993.

[10] R. A. Jacobs. Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*, 1:295--307, 1988.

[11] H. Kersten. Accelerated Stochastic Approximation. *Annals of Mathematical Statistics*, 29:41--59, 1958.

[12] A. N. Kolmogorov. Three Approaches to the Quantitative Definition of Information. *Problems of Information Transmission*, 1(1): 1--7, Jan. - Mar. 1965. Faraday Press translation of Problemy Peredachi Informatsii.

[13] S. Kullback and A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:79--86, 1951.

[14] Y. LeCun, J. Denker, and S. Solla. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems, vol.* 2, pages 598--605. Morgan Kauffman, San Mateo, CA, 1990.

15] B. A. Pearlmutter. Fast Exact Multiplication by the Hessian. *Neural Computation*, 6(1): 147--160, January 1994.

16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning Representations by Backpropagation Errors. *Nature*, 323:533--536, October 1986.

17] D. E. Rumelhart, J. L. McClelland, et al. *Parallel Distributed Processing*, volume 1. MIT Press, 1987.

18] R. S. Sutton. Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta. in *Proceedings of the AAAI*, pages 171 -- 176, 1992.